

Optimal Algorithms for Local Vertex Quartet Cleaning*

Gianluca Della Vedova
Dipartimento di Statistica
Università degli Studi di Milano–Bicocca
Milano, Italy.

gianluca.dellavedova@unimib.it

H. Todd Wareham
Department of Computer Science
Memorial University of Newfoundland
St. John's, NF, Canada A1B 3X5.

harold@cs.mun.ca

ABSTRACT

Reconstructing evolutionary trees is an important problem in biology. A response to the computational intractability of most of the traditional criteria for inferring evolutionary trees has been a focus on new criteria, particularly quartet-based methods that seek to merge trees derived on subsets of four species from a given species-set into a tree for that entire set. Unfortunately, most of these methods are very sensitive to errors in the reconstruction of the trees for individual quartets of species. A recently-developed technique called *quartet cleaning* can alleviate this difficulty in certain cases by using redundant information in the complete set of quartet topologies for a given species-set to correct such errors. In this paper, we describe two new local vertex quartet cleaning algorithms which have optimal time complexity and error-correction bound, respectively. These are the first known local vertex quartet cleaning algorithms that are optimal with respect to either of these attributes.

Keywords

evolutionary trees, quartet-based methods, algorithms

1. INTRODUCTION

Recent advances in Molecular Biology [4] have resulted in huge amounts of biological sequence data over a steadily increasing number of species. One application of this data is the derivation of hypotheses about how a given group of species evolved from their most recent common ancestor. Each such hypothesis is typically represented by an *evolutionary tree* or *phylogeny*, which is an edge-weighted rooted tree such that the leaves are labeled by sequences of given species, the internal nodes represent the (possibly unknown) sequences associated with ancestral species, and the weights on the edges represent the evolutionary distances

*Most of the work reported here was done while the first author was visiting the Department of Computing and Software at McMaster University, Hamilton, ON, Canada.

between species. In theory, larger datasets are desirable because they seem to result in more accurate trees; however, in practice, large datasets make exhaustive tree-evaluation methods such as Maximum Likelihood and Maximum Parsimony unusable (see [12] and references).

In order to overcome the computational difficulties associated with traditional phylogeny inference methods, a number of *quartet based* methods have been proposed in recent years [1, 2, 5, 8, 10, 11]. A *quartet* is a set of four species and a *quartet topology* is a partition of a quartet into two subsets, where in each set there are the two species that are more closely related. The quartet paradigm divides the phylogeny inference process into two steps: the first step is, given the species sequences, applying a *quartet topology inference* method [10, 12] to reconstruct the (supposed) quartet topologies, the second step is, given all quartet topologies, applying a *quartet recombination* method [2, 8, 11] to reconstruct an evolutionary tree topology. At the end of this two-step process the tree is rooted and weights are assigned to the edges.

Most research has focused on the latter step. However, the former step is critical as almost all quartet-recombination methods fail if even a small proportion of the reconstructed quartet topologies are erroneous. Indeed, the problem of erroneous quartet topologies is considered one of the main drawbacks of quartet-based methods.

One approach to dealing with erroneous quartet topologies is to ask for the tree that is consistent with the largest possible set of derived quartets. Unfortunately, the decision version of this problem is *NP*-complete [3] and hence is unlikely to be solvable efficiently. A much more interesting alternative has been proposed under the title of quartet cleaning [9]. The main idea is to modify the two-step process described above to allow an intermediate step that can detect and correct *all* erroneous quartet topologies, *provided the distribution of these errors in the phylogeny is sparse*. This is possible because each quartet topology error affects a restricted portion of the tree – namely, those vertices or edges in the tree such that the quartet is across that edge or vertex (see next section for a formal definition). Quartet cleaning techniques can correct all errors across an edge or vertex provided that a number of errors is bounded by fixed fraction of the total number of quartets across that edge or vertex. The quartet cleaning algorithms presented to date can be split into two classes: global and local. The former assume that the bound on the number of errors holds for all vertices (edges) of the tree and only reconstructs the tree in that case, while the latter only require that the bound

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC 2002, Madrid, Spain

Copyright 2002 ACM 1-58113-445-2/02/03 ...\$5.00.

hold for some vertices (edges) and reconstructs the portion of the tree consisting of all vertices (edges) for which the error bound is satisfied.

In this paper, we will describe two optimal local vertex cleaning algorithms; the first has an optimal time complexity that is linear in the size of the given set of all quartet topologies but has a sub-optimal error correction bound, while the second has an almost quadratic time complexity but is optimal with respect to the number of errors corrected. These algorithms compare favorably with the only previously described local vertex cleaning algorithm [3] which is suboptimal with respect to both the time complexity and the number of errors that can be corrected. Furthermore, both of these algorithms are relatively simple and derive their power from new analyses of the relationship between the error bound on a vertex and the number of correct quartet topologies that must be across that vertex.

2. TERMINOLOGY

In this paper, we will denote the given species set by S , and let n be the cardinality of S , *i.e.*, $n = |S|$. An *evolutionary tree* T over S is a tree whose leaves are exactly the members of S and whose internal nodes have degree 3. A *quartet* from S is any subset of S of 4 elements. A *quartet topology* is a partition of a quartet into two subsets of two elements each, written as $ab|cd$. The quartet topology (or resolution) *induced* by a tree T for a quartet $\{a, b, c, d\}$ is $ab|cd$ if and only if a is closer to b than either c or d in T . For example, the quartet topologies induced by the tree T in Figure 1 for $\{a, b, c, d\}$, $\{c, f, g, h\}$, and $\{f, g, h, i\}$ are $ab|cd$, $cf|gh$, and $fg|hi$. In the following, Q_T will denote the set of all possible quartet topologies induced by T .

Given an evolutionary tree on a species-set S , each internal node v of T induces a *tripartition* (A_v, B_v, C_v) of S such that $T - \{v\}$ consists of three trees whose set of leaves are A_v , B_v , and C_v . The tripartition induced by vertex v in T is shown in Figure 1. Let $Q(A_v, B_v, C_v)$ (or $Q_T(v)$) denote

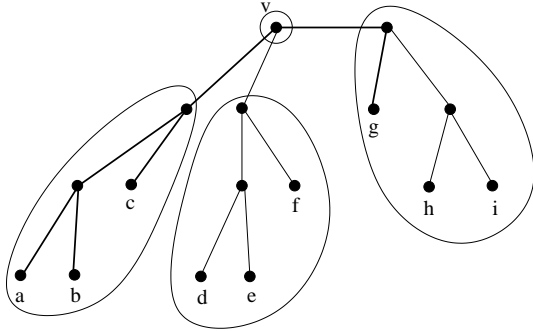


Figure 1: Tripartition induced by v in T .

the set of quartets $\{a, b, c, d\}$ such that $a \in A_v$, $b \in B_v$, and $c \in C_v$: each such quartet is said to be *across the vertex* v . For example, in the tree T in Figure 1, the quartet $\{a, b, f, i\}$ is across vertex v but quartet $\{a, b, c, g\}$ is not across vertex v . Similarly, each edge e of T induces a bipartition (A_e, B_e) such that removing the edge e from T gives two trees with leaves A_e and B_e respectively. A quartet $\{a, b, c, d\}$ is said to be *across the edge* e (or across $\langle A_e, B_e \rangle$) if and only if $|A_e \cap \{a, b, c, d\}| = 2$. Two tripartitions (bipartitions) are

compatible if there exists a phylogeny that induces both of them. An efficient algorithm for computing the tree inducing a given set of compatible bipartitions is given in [7]; note that this algorithm can be easily extended to the case where a set of tripartitions is given as input. Let Q be a complete set of quartet topologies over S (that is the resolutions for all possible quartets over S) and let T be a phylogeny over S . A quartet $\{a, b, c, d\}$ over S is a *quartet error* for T if its resolution in Q is different from that in Q_T . The number of quartets across a vertex v is $\frac{1}{2}(|S| - 3)|A_v||B_v||C_v|$; this is also an upper bound for the number of quartet errors across a vertex.

3. QUARTET CLEANING ALGORITHMS

	Local	Global
edge	2-bounded , $O(n^5)$ time [6]	2-bounded , $O(n^4)$ time [3]
vertex	4-bounded, $O(n^7)$ time [3] 9-bounded, $O(n^4)$ time (*) 2-bounded , $O(n^7)$ time (*)	–

Table 1: Quartet Cleaning Algorithms: Known Results. Boldface means optimality of an attribute is denoted by boldfacing that attribute. Algorithms developed in this paper are starred (*).

Informally, a quartet cleaning algorithm [3] computes a phylogeny consisting of vertices or edges such that the number of quartet errors across those vertices or edges is bounded by a given value. The following definitions formalize this description. Let T be an evolutionary tree over a set S , and let $\langle A, B \rangle$ be a bipartition of S . Then $\langle A, B \rangle$ is α -bounded if the number of quartet errors across such bipartition is strictly less than $(|A| - 1)(|B| - 1)/\alpha$. Let T be an evolutionary tree over a set S , and let $\langle A, B, C \rangle$ be a tripartition of S . Then $\langle A, B, C \rangle$ is called α -bounded if all bipartitions $\langle A, B \cup C \rangle$, $\langle B, A \cup C \rangle$, $\langle C, A \cup B \rangle$ are α -bounded.

A quartet cleaning algorithm has *global vertex (edge) bound* α if it corrects all quartet errors in phylogenies such that the tripartition (bipartition) induced by each vertex (edge) is α -bounded, and a quartet cleaning algorithm has *local vertex (edge) bound* α if it corrects all quartet errors across any vertex (edge) whose induced tripartition (bipartition) is α -bounded [3]. Note that a local vertex algorithm does not require the bound to hold for all vertices. Hence such algorithms are more robust than global cleaning algorithms

AssociatePartition(S, Q, a, b, c)

Input: A triplet $\{a, b, c\}$ from the species-set S .

Output: A tripartition that is compatible with the given triplet relative to Q .

$A := \{a\}; B := \{b\}; C := \{c\}$.

For each $s \in S - \{a, b, c\}$ **Do**

If $as|bc \in Q$ **Then** $A := A \cup \{s\}$

If $sb|ac \in Q$ **Then** $B := B \cup \{s\}$

If $ab|cs \in Q$ **Then** $C := C \cup \{s\}$

EndFor

Return the tripartition $\langle A, B, C \rangle$

Table 2: Algorithm AssociatePartition.

that require the bound to hold for all edges or vertices of the tree.

The state of the art in quartet cleaning algorithms is summarized in Table 1. The idea of quartet cleaning was first described in [9]. This paper gave a polynomial-time global edge cleaning algorithm with bound 2. This bound is asymptotically optimal, as there exist examples where $(|A_e| - 1)(|B_e| - 1)/2$ quartet errors across an edge e with associated bipartition (A_e, B_e) make it impossible for any algorithm to correct all quartet errors across that edge. Unfortunately, the polynomial in the time complexity is of very high order and the algorithm is of theoretical interest only. A 2-bounded global edge cleaning algorithm and a 4-bounded local vertex cleaning algorithm were given in [3]; the time complexity of the former is linear in the number of given quartet topologies *i.e.*, $O(n^4)$, and is hence optimal, while the latter runs in $O(n^7)$ time. A 2-bounded local edge cleaning algorithm that runs in $O(n^5)$ time was subsequently described in [6] in the context of a more accurate (albeit more complex) quartet cleaning framework called *hypercleaning* (see [6] for details).

In the following two sections, we will describe a α -bounded local vertex cleaning algorithm that runs in $O(n^4)$ time for $\alpha \geq 9$ and a 2-bounded local vertex cleaning algorithm that runs in $O(n^7)$ time. Both of the algorithms exploit the Theorem 3 in [3]: Any two 2-bounded tripartitions of a set S , $\langle A_1, B_1, C_1 \rangle$ and $\langle A_2, B_2, C_2 \rangle$ are compatible.

The proof given in [3] is for the case of 4-bounded tripartitions, but it holds also for the case of α -bounded tripartitions, with $\alpha \geq 2$. Hence, to describe a local vertex cleaning algorithm, it suffices to compute a set of 2-bounded tripartitions containing all α -bounded tripartitions.

All algorithms described in this paper rely on a procedure `AssociatePartition` which associates a tripartition with a given triplet of species $a, b, c \in S$ (see Table 2). Given a tripartition $\langle A, B, C \rangle$ associated with the triplet $\{a, b, c\}$ we will say that all quartets $\{a, b, c, s\}$ with $s \in S - \{a, b, c\}$ *witness* the tripartition $\langle A, B, C \rangle$. Whenever we mention partitions witnessed by some quartets, we mean partitions computed by `AssociatePartition`. Clearly if the number of witnesses of a tripartition is large, then it is likely that such tripartition is correct. In particular if there are no quartet errors in Q then the set of witnesses of a tripartition $\langle A, B, C \rangle$ is exactly the set of quartets across $\langle A, B, C \rangle$, which contains $|A||B||C|(|S| - 3)/2$ elements.

LEMMA 3.1. *Let (a, b, c, d) be a quartet of species in S . Then (a, b, c, d) may witness at most 4 tripartitions.*

PROOF. By construction. Only tripartitions associated with a triplet in the set $\{a, b, c, d\}$ can be witnessed by the quartet $\{a, b, c, d\}$. The claim follows as there are exactly 4 such triplets. \square

Note that Lemma 3.1 implies that there are at most $O(n^4)$ tripartitions that have at least one witness.

LEMMA 3.2. *Let $\langle A, B, C \rangle$ be a tripartition of the set S induced by the removal of a vertex in the tree, and (a, b, c, s) be a quartet error with $a, s \in A$, $b \in B$, $c \in C$. Then at most $2|S| - 7$ missing witnesses of $\langle A, B, C \rangle$ are due to such error.*

Lemma 3.2, whose proof is omitted, suggests that an α -bounded tripartition must have a sufficiently large number of witnesses. In the next section we will prove that result for the case of $\alpha \geq 9$.

4. A TIME-OPTIMAL ALGORITHM

In this section, we describe a $O(n^4)$ time algorithm that computes all α -bounded tripartitions associated with a given complete set of quartet topologies for S when $\alpha \geq 9$. As this set of quartet topologies is of size $O(n^4)$, this time complexity is optimal. The algorithm exploits the fact that α -bounded tripartitions must have a certain number of witnesses. The following lemma establishes the relationship between α and the number of witnesses. The proof is rather technical and is omitted due to page constraints.

LEMMA 4.1. *Let α, e be two constants such that $\alpha \geq 9$, $e \geq \frac{4\alpha}{\alpha-8}$ and let $\langle A, B, C \rangle$ be an α -bounded tripartition with $|A| + |B| + |C| \geq (\frac{12}{\alpha} - \frac{3e-6}{2e})e$. Then $\langle A, B, C \rangle$ have at least $|A||B||C|(|S| - 3)/e$ witnesses.*

For simplicity's sake our algorithm assumes that $n \geq e(12/\alpha + \frac{3e-6}{2e})$, where e satisfies the hypothesis of Lemma 4.1, otherwise we have only a constant number of species and all bipartitions can be checked for α -boundedness.

The first part of a local vertex cleaning algorithm that exploits this property is given in Table 3. The algorithm begins by computing the tripartition associated with each triplet $\{a, b, c\}$. In a convenient abuse of notation, `AssociatePartition` (S, Q, a, b, c) also denotes the partition returned by algorithm `AssociatePartition`. Note that after this first step is completed, it is possible to access `AssociatePartition` (S, Q, a, b, c) in constant time. We assume an ordering s_1, \dots, s_n of the elements in S so we can represent a tripartition with a sequence of n integers $\langle p_1, \dots, p_n \rangle$ where $p_i = 1$ if $s_i \in A$, $p_i = 2$ if $s_i \in B$, and $p_i = 3$ if $s_i \in C$. For example, the tripartition $\{a_1, a_4\}, \{a_2, a_5\}, \{a_3\}$ of $\{a_1, a_2, a_3, a_4, a_5\}$ is stored as $\langle 1, 2, 3, 1, 2 \rangle$ and the partition $\{\{a_1, a_2\}, \{a_3, a_5\}, \{a_4\}\}$ is stored as $\langle 1, 1, 2, 3, 2 \rangle$. This encoding allows the examination of the elements of a partition in $O(n)$ time. Note further that by Lemma 3.1, each quartet can witness at most 4 tripartitions and our algorithm associates each tripartition with some set of triplets. Information about the tripartition-quartet relationship will be stored in an array of lists $witness(q)$ indexed by quartets of S , where $witness(q) = \{(a, b, c) : q \text{ witnesses } \text{AssociatePartition}(S, Q, a, b, c)\}$.

Consider the time complexity of algorithm `ConstructPartitions`. The key observation for proving that Part 1 has $O(n^4)$ time complexity is that each quartet witnesses at most 4 tripartitions; hence inserting into set $witness(a, b, c, s)$ can be done in constant time. Since each call to `AssociatePartition` requires $O(n)$ time, the total time spent in Part 1 is $O(n^4)$. Computing the number of witnesses of a tripartition p can be done in $O(w)$ time, where w is the number of witnesses of p , as the witnesses must appear consecutively in P_T . By Lemma 3.1, the total number of witnesses (and consequently the time required in Part 2) is $O(n^4)$. As for Part 3, note that the set P prior to that loop contains only those tripartitions p such that the number of witnesses of p is at least $|A||B||C|(|S| - 3)/e$ times the number of quartets across the vertex associated with p . By Lemma 3.2, the total number of quartets across partitions that need to be examined in that loop is $O(n^4)$ and this final loop runs in $O(n^4)$ time. This leads to an overall $O(n^4)$ time complexity for algorithm `ConstructPartitions`.

To complete the local vertex cleaning algorithm, note that by Lemma 3.1, there is a tree compatible with the tripartitions in P , and moreover, this tree can be computed in

$O(n^4)$ time by a standard algorithm [7]. Hence, the algorithm as a whole runs in $O(n^4)$ time.

```

ConstructPartitions( $S, Q, \alpha$ )
Input: A complete set  $Q$  of quartet topologies over the
species-set  $S$ , with  $|S| = n$ .
Output: The list  $P$  of compatible  $\alpha$ -bounded tripartitions
of  $S$  induced by  $Q$ .
/* Part 1 */
Let  $P_T$  be an empty table.
For each triplet  $a, b, c \in S$  Do
   $p := \text{AssociatePartition}(S, Q, a, b, c)$ 
  For each  $s \in S - \{a, b, c\}$  Do
    If  $p \notin \text{witness}(a, b, c, s)$  Then
      Add entry  $\langle (a, b, c, s), p \rangle$  to  $P_T$ .
       $\text{witness}(a, b, c, s) := \text{witness}(a, b, c, s) \cup \{p\}$ 
    EndIf
  EndFor
EndFor
/* Part 2 */
Sort  $P_T$  using the partitions as primary key and the
quartets as secondary key.
For each stream  $t_i, \dots, t_{i+k}$  of consecutive rows of  $P_T$ 
with the same partition  $p$  Do
  Add  $p$  to the set  $P$  of computed tripartitions
  Compute the number of witnesses of  $p$ .
EndFor
/* Part 3 */
For each tripartition  $p = \langle A, B, C \rangle \in P$  Do
  If  $|\text{witness}(p)| < |A||B||C|(n-3)/e$  then
    remove  $p$  from  $P$ .
  EndFor
For each partition  $p = \langle A, B, C \rangle \in P$  Do
  Remove  $p$  from  $P$  if  $p$  is not 2-bounded
EndFor
Return  $P$ 

```

Table 3: Algorithm ConstructPartitions.

5. AN ERROR-OPTIMAL ALGORITHM

In this section, we describe a $O(n^7)$ time algorithm that computes all 2-bounded tripartitions associated with a given complete set of quartet topologies for S . As there exist examples where $(|A_e| - 1)(|B_e| - 1)/2$ quartet errors across an edge e with associated bipartition (A_e, B_e) make it impossible for any algorithm to correct all quartet errors across that edge [9], this error bound is optimal.

The algorithm described in this section exploits some particularly convenient restrictions on the numbers of witness that must exist for any 2-bounded tripartition. These restrictions are best appreciated in the context of the simple local vertex cleaning algorithm given in Table 4. Whenever there exists a triplet $\{a, b, c\}$ with $p = \text{AssociatePartition}(S, Q, a, b, c)$, algorithm **CleanSimple** is guaranteed to output a list L of partitions including p . Consequently we only have to ensure that we also include the 2-bounded partitions that are not associated with any triplet, which means that we now have to deal only with the case where all tripartitions associated with the triplets $a \in A$, $b \in B$, and $c \in C$ are incorrect.

```

CleanSimple( $S, Q$ )
Input: A complete set  $Q$  of quartet topologies over the
species-set  $S$ .
Output: The list  $P$  of compatible 2-bounded tripartitions
of  $S$  induced by  $Q$ .
 $P := \emptyset$ 
For each triplet  $a, b, c \in S$  Do
   $p := \text{AssociatePartition}(S, Q, a, b, c)$ 
  If  $p$  is 2-bounded Then
     $P := P \cup \{p\}$ 
  EndIf
EndFor
Return  $P$ 


---


CorrectError( $S, Q, \langle A, B, C \rangle$ )
Input: A tripartition  $\langle A, B, C \rangle$  of species-set  $S$  such
at least one of  $A$ ,  $B$ , and  $C$  has less than 6 elements.
Output: A list  $P$  of tripartitions, one of which is the
cleaned version of the input tripartition.
 $P := \emptyset$ 
For each  $s \in S$  Do
   $A' := A - \{s\}; B' := B - \{s\}; C' := C - \{s\};$ 
   $P := P \cup \{\langle A' \cup \{s\}, B', C' \rangle\}$ 
   $P := P \cup \{\langle A', B' \cup \{s\}, C' \rangle\}$ 
   $P := P \cup \{\langle A', B', C' \cup \{s\} \rangle\}$ 
EndFor
For each partition  $p \in P$  Do
  Remove  $p$  from  $P$  if  $p$  is not 2-bounded
EndFor
Return  $P$ 


---


CleanRevised( $S, Q$ )
Input: A complete set  $Q$  of quartet topologies over the
species-set  $S$ .
Output: The list  $P$  of compatible 2-bounded tripartitions
of  $S$  induced by  $Q$ .
 $P := \emptyset$ 
For each triplet  $a, b, c \in S$  Do
   $p := \text{AssociatePartition}(S, Q, a, b, c)$ 
  If  $|A| < 6$  or  $|B| < 6$  or  $|C| < 6$  Then
     $P := P \cup \text{CorrectError}(S, Q, p)$ 
  Else
     $P := P \cup \{p\}$ 
  EndIf
EndFor
For each partition  $p = \langle A, B, C \rangle \in P$  Do
  Remove  $p$  from  $P$  if  $p$  is not 2-bounded
EndFor
Return  $P$ 

```

Table 4: Algorithms CleanSimple, CleanRevised and CorrectError.

Consider a 2-bounded tripartition $\langle A, B, C \rangle$. By Definition the number of quartet errors across such a tripartition is less than $\frac{1}{2}(2|A||B| + 2|A||C| + 2|B||C| - 3|A| - 3|B| - 3|C| + 3)$. Let $\text{err}(a, b, c)$ be the set $(Q_T - Q) \cap \{\{a, b, c, w\} : w \in S - \{a, b, c\}\}$. Intuitively, $\text{err}(a, b, c)$ consists of all quartet errors among the quartets considered inside a call to algorithm **AssociatePartition**(S, Q, a, b, c). Since a single quartet error can only influence the tripartitions associated with two distinct triplets [3, Theorem 2], for a 2-bounded tripartition $\langle A, B, C \rangle$ the sum of $|\text{err}(a, b, c)|$ over all triplets $\{a, b, c\}$ with $a \in A$, $b \in B$, and $c \in C$ is less than $2|A||B| + 2|A||C| +$

$2|B||C| - 3|A| - 3|B| - 3|C| + 3$. Since the number of triplets is equal to $|A||B||C|$, we the average number of errors per triplet is $\frac{2|A||B|+2|A||C|+2|B||C|-3|A|-3|B|-3|C|+3}{|A||B||C|}$

Note that whenever this average value is strictly less than an integer avg , there is a triplet (a, b, c) whose associated tripartition has at most $avg - 1$ quartet errors across it.

We first show that any 2-bounded tripartition has at most one quartet error across its associated vertex. If $|C| = 1$ then the number of errors is $2|A||B| - |A| - |B|$, and $\frac{2|A||B|-|A|-|B|}{|A||B|} < 2$. Similarly, if $|C| = 2$, $\frac{2|A||B|+|A|+|B|-3}{2|A||B|} < 2$. Therefore, if $|C| < 3$, there is a triplet whose associated tripartition has at most one error. As the above inequalities hold for all choices of $|A|$ and $|B|$, this also holds if $|A| < 3$ or $|B| < 3$ by symmetry. We can also prove that the average number of errors is less than 2 in the remaining case when $|A| \geq 3$, $|B| \geq 3$, and $|C| \geq 3$.

Since $|A|, |B|$, and $|C|$ are all greater than or equal to three, the left hand side is not greater than zero, while the right hand side is strictly positive.

Now assume that all sets A, B, C contain at least six elements. In this case, we can prove that the average number of errors is less than one, *i.e.*, zero: $2|A||B| + 2|A||C| + 2|B||C| - 3|A| - 3|B| - 3|C| + 3 < |A||B||C| \Rightarrow |A||B|(2 - \frac{|C|}{3}) + |A||C|(2 - \frac{|B|}{3}) + |B||C|(2 - \frac{|A|}{3}) < 3|A| + 3|B| + 3|C| - 3$

Just as in the previous case, as $|A|, |B|, |C|$ are all greater than or equal to 6, the left hand side cannot be greater than zero while the right hand side is strictly positive.

Two important consequences of the analysis given above are (1) all tripartitions $\langle A, B, C \rangle$ such that all sets contain at least 6 elements are already computed by the procedure `CleanSimple` described at the beginning of this section, and (2) each of the remaining 2-bounded tripartitions has at most one error across its associated vertex. A brute force algorithm `CorrectError` that corrects all such errors and returns a list of tripartitions that includes the correct tripartition, together with a revised cleaning algorithm, is given in Table 4.

Consider the time complexity of algorithm `CleanRevised`. The most expensive step in this algorithm is checking all tripartitions for 2-boundedness. As the algorithm computes $O(n^3)$ tripartitions for each triplet and, as described in the previous section, each such tripartition can be checked in $O(n^4)$ time, $O(n^7)$ time suffices to check all tripartitions. A potential problem may be introduced by the additional set of $O(n)$ tripartitions that may be computed by algorithm `CorrectError` for each tripartition. This would appear to raise the number of tripartitions that must be checked to $O(n^4)$. However, these tripartitions are not completely general, as each of the tripartitions given as input to `CorrectError` have one set that contains at most 6 elements; this implies that the tripartitions returned as output differ from the input tripartition by at most one element, which in turn implies that all tripartitions returned by `CorrectError` have one set containing at most 7 elements. Consequently, the number of quartets across a tripartition returned by `CorrectError` is $O(n^3)$, and algorithm `CleanRevised` can in fact compute all 2-bounded tripartitions in $O(n^7)$ time.

6. CONCLUSIONS

In this paper, we have described two local vertex quartet cleaning algorithms, the first of which has an optimal (linear) running time but is guaranteed to recover less than

one fourth of the optimal number of errors, while the second recovers an optimal number of errors but has an almost quadratic running time. An important open problem is determining whether or not there is a local vertex cleaning algorithm that is optimal in terms of both running time and recovered errors.

7. REFERENCES

- [1] A. Ben-Dor, B. Chor, D. Graur, R. Ophir, and D. Pelepp. Constructing Phylogenies from Quartets: Elucidation of Eutherian Superordinal Relationships. *Journal of Computational Biology* 5(3): 377-390, 1998.
- [2] V. Berry and O. Gascuel. Inferring evolutionary trees with strong combinatorial evidence. *Theor. Comp. Sci.* 240(2): 271-298, 2000.
- [3] V. Berry, T. Jiang, P. E. Kearney, M. Li, and T. Wareham. Quartet cleaning: Improved algorithms and simulations. In *7th European Symposium on Algorithms (ESA '99)*. Volume 1643 of LNCS, 313-324, 1999.
- [4] M. Boguski. Bioinformatics - a new era. In S. Brenner, F. Lewitter, M. Patterson, and M. Handel, editors, *Trends Guide to Bioinformatics*. Elsevier Science. 1-3, 1998.
- [5] D. Bryant. *Building Trees, Hunting for Trees, and Comparing Trees*. Ph.D. thesis, University of Canterbury. 1997.
- [6] D. Bryant, V. Berry, P. Kearney, M. Li, T. Jiang, H.T. Wareham, and H. Zhang. A Practical Algorithm for Recovering the Best Supported Edges of an Evolutionary Tree. In *11th ACM-SIAM Symposium on Discrete Algorithms (SODA 2000)*. 287-296, 2000.
- [7] P. Buneman. The recovery of trees from measures of dissimilarity. In *Mathematics in the Archaeological and Historical Sciences*. Edinburgh University Press. 1971.
- [8] P. Erdős, M. Steel, L. A. Székely, and T. Warnow. Constructing big trees from short sequences. In *24th International Colloquium on Automata, Languages and Programming (ICALP'97)*. Volume 1256 of LNCS. 827-837. 1997
- [9] T. Jiang, P. Kearney, and M. Li. A Polynomial Time Approximation Scheme for Inferring Evolutionary Trees from Quartet Topologies and Its Application. *SIAM J. Comput.* 30(6): 1942-1961. 2000
- [10] P. Kearney. The ordinal quartet method. In *2nd Annual International Conference on Computational Molecular Biology (RECOMB'98)*. 125-134. 1998.
- [11] K. Strimmer and A. von Haeseler. Quartet puzzling: A quartet maximum-likelihood method for reconstructing tree topologies. *Mol. Biol. Evol.*, 13(7), 964-969. 1996.
- [12] D.L. Swofford, G.J. Olsen, P.J. Waddell, and D.M. Hillis. Phylogenetic Inference. In D.M. Hillis, C. Moritz, and B.K. Mable (eds.) *Molecular Systematics* Sinauer Assoc. Sunderland, MA. 407-514. 1996.